APPENDIX

EXAMPLE

Here is a table that summarizes the models expected in Listing 2.

M^+	M^-	$\mathcal{D}_X, \mathcal{D}_Y$
{ a, d }	{ b, e, f, g, h }	{1},{1}
{ a, g }	{ b, d, e, f, h }	{1},{2}

Here is a table that summarizes the two families of models expected in Example 1 The first family of answer is given by : { `physics != 1, math < mathLab, physics = 2, biology = 1, mathLab = 2, math = 1` } The second one : { `physics != 1, math < mathLab, physics = 2, biology = 2, mathLab = 2, math = 1` } It's quite intuitive to see that the second model does not have any solution. Three classes takes place in the afternoon and there is only two rooms. Since there is one and only one class per room, by the pigeonhole principle we can tell that there is no solution to this model. On the other hand, the first model leads to two possible solutions. Since physics takes place in the room a, mathLab must take place in room b. In the morning, math and biology can permute leaving us with two possible assignment.

The two solutions to this program are : { assignement(physics,2,a), assignement(biology,1,a), assignement(math,1,b), assignement(mathLab,2,b) and { assignement(physics,2,a), assignement(biology,1,b), assignement(math,1,a), assignement(mathLab,2,b) } with the valuation { `physics != 1, math < mathLab, physics = 2, biology = 1, mathLab = 2, math = 1` }.

PROOFS (FOR REVIEWING PURPOSES)

Proof of Lemma 1. Let $P_1 = (\mathcal{R}_1, \mathcal{R}_C, \mathcal{V}, \mathcal{D})$ and $P_2 = (\mathcal{R}_2, \mathcal{R}_C, \mathcal{V}, \mathcal{D})$ and $\mathcal{R}_2 = \mathcal{R}_1 \setminus \{r\}$ with $r \in \mathcal{R}_1$. Then by definition $P_1 \preccurlyeq P_2$. The reductions $P_1 \overset{*}{\Longrightarrow} P_2$ with $* \in \{\mathtt{N}, \mathtt{L}, \mathtt{F}\}$ implies that a rule is removed from the program and thus $P_1 \preccurlyeq P_2$.
Let $P_1 = (\mathcal{R}_1, \mathcal{R}_C, \mathcal{V}, \mathcal{D})$ and $P_2 = (\mathcal{R}_2, \mathcal{R}_C, \mathcal{V}, \mathcal{D})$ with $r \in \mathcal{R}_1$ and $r' \in \mathcal{R}_2$. By definition $\mathtt{body}(r) = \mathtt{body}(r') \setminus \{b\}$. The rule r has been simplified, thus $r \preccurlyeq r'$. Then by definition $P_1 \preccurlyeq P_2$. The reductions $P_1 \overset{*}{\Longrightarrow} P_2$ with $* \in \{\mathtt{P}, \mathtt{S}\}$ implies that a rule is simplified in the program and thus $P_1 \preccurlyeq P_2$.

Let $P_1 = (\mathcal{R}, \mathcal{R}_{C1}, \mathcal{V}, \mathcal{D})$ and $P_2 = (\mathcal{R}, \mathcal{R}_{C2}, \mathcal{V}, \mathcal{D})$ and $\mathcal{R}_{C2} = \mathcal{R}_{C1} \setminus \{r\}$ with $r \in \mathcal{R}_{C1}$. Then by definition $P_1 \preccurlyeq P_2$. The reductions $P_1 \overset{*}{\Longrightarrow} P_2$ with $* \in \{\mathtt{N}, \mathtt{L}, \mathtt{F}\}$ implies that a rule is removed from the program and thus $P_1 \preccurlyeq P_2$.
Let $P_1 = (\mathcal{R}, \mathcal{R}_{C1}, \mathcal{V}, \mathcal{D})$ and $P_2 = (\mathcal{R}, \mathcal{R}_{C2}, \mathcal{V}, \mathcal{D})$ with $r \in \mathcal{R}_{C1}$ and $r' \in \mathcal{R}_{C2}$. By definition $\mathtt{body}(r) = \mathtt{body}(r') \setminus \{b\}$. The rule r has been simplified, thus $r \preccurlyeq r'$. Then by definition $P_1 \preccurlyeq P_2$. The reductions $P_1 \overset{*}{\Longrightarrow} P_2$ with $* \in \{\mathtt{P}, \mathtt{S}\}$ implies that a rule is simplified in the program and thus $P_1 \preccurlyeq P_2$.

Proof of Lemma 2. A CASP program is represented as a triplet, with reductions exclusively operating on regular atoms. These reductions do not directly modify the domains.

Proof of Theorem 2. For this proof, we will consider that each program has a set of rule $\mathcal{R}$ containing the of rules with constraints in their heads $\mathcal{R}_C$. First, let's prove that every pair of reduction are semi-confluent. The subsequent proof establishes that constraint success reduction and failure reduction are semi-confluent by proving that any divergent reductions from a common origin can eventually be reconciled to a single resulting state.Let $\mathcal{R}$ be a set of rules such that c_x is a satisfied constraint and $y \notin \mathtt{heads}(\mathcal{R})$.

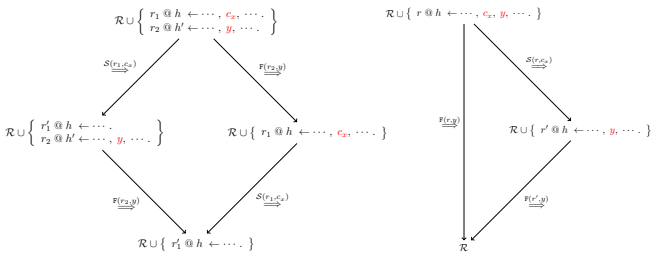

Fig. 1. Confluence of the constraint success reduction and failure reduction.

The subsequent proof establishes that constraint success reduction and negative reduction are semi-confluent by proving that any divergent reductions from a common origin can eventually be reconciled to a single resulting state.Let $\mathcal{R}$ be a set of rules such that c_x is a satisfied constraint and y is a fact.

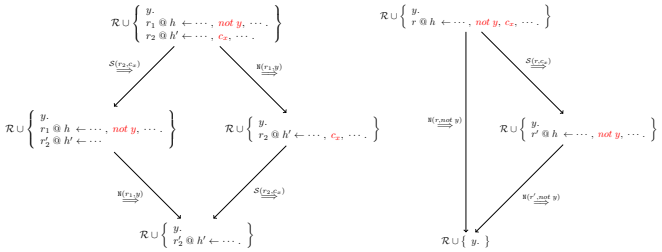

Fig. 2. Confluence of the constraint success reduction and negative reduction.

The subsequent proof establishes that constraint success reduction and failure reduction are semi-confluent by proving that any divergent reductions from a common origin can eventually be reconciled to a single resulting state.Let $\mathcal{R}$ be a set of rules such that c_x is a constraint that is unsatisfiable and c_y is a satisfied constraint.

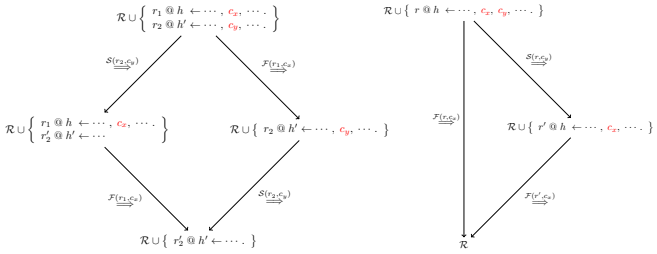

Fig. 3. Confluence of the constraint success reduction and constraint failure reduction.

The subsequent proof establishes that constraint success reduction and positive reduction are semi-confluent by proving that any divergent reductions from a common origin can eventually be reconciled to a single resulting state.Let $\mathcal{R}$ be a set of rules such that c_x is a satisfied constraint and $y \notin \texttt{heads}(\mathcal{R})$.

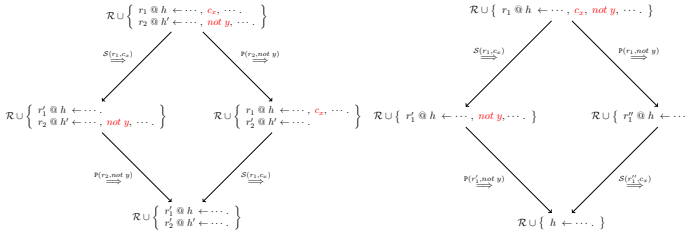

Fig. 4. Confluence of the constraint success reduction and positive reduction.

The subsequent proof establishes that constraint success reduction and success reduction are semi-confluent by proving that any divergent reductions from a common origin can eventually be reconciled to a single resulting state.Let $\mathcal{R}$ be a set of rules such that c_x is a satisfied constraint and $y \notin \texttt{heads}(\mathcal{R})$.

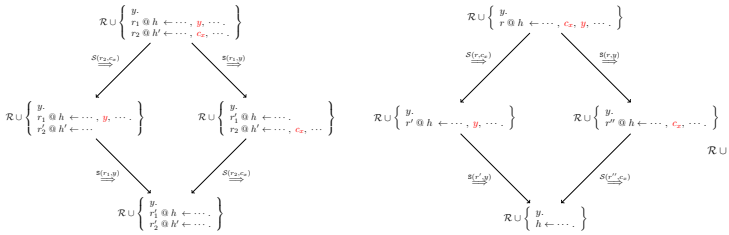

Fig. 5. Confluence of the constraint success reduction and success reduction.

The subsequent proof establishes that constraint failure reduction and failure reduction are semi-confluent by proving that any divergent reductions from a common origin can eventually be reconciled to a single resulting state. Let $\mathcal{R}$ be a set of rules such that c_x is an unsatisfiable constraint and $y \notin \texttt{heads}(\mathcal{R})$.

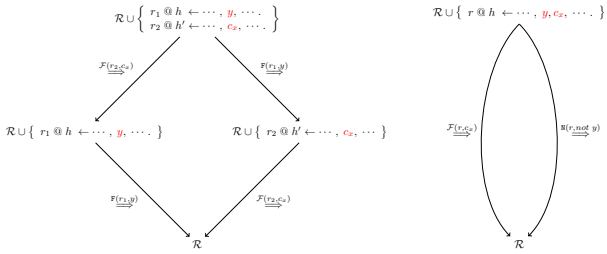

Fig. 6. Confluence of the constraint failure reduction and failure reduction.

The subsequent proof establishes that constraint failure reduction and negative reduction are semi-confluent by proving that any divergent reductions from a common origin can eventually be reconciled to a single resulting state. Let $\mathcal{R}$ be a set of rules such that c_x is a satisfied constraint and $y \notin \texttt{heads}(\mathcal{R})$.

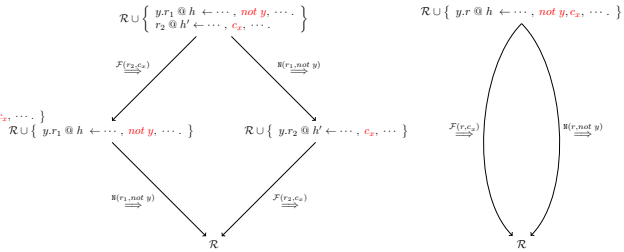

Fig. 7. Confluence of the constraint failure reduction and negative reduction.

The subsequent proof establishes that constraint failure reduction and positive reduction are semi-confluent by proving that any divergent reductions from a common origin can eventually be reconciled to a single resulting state. Let $\mathcal{R}$ be a set of rules such that c_x is a satisfied constraint and $y \notin \texttt{heads}(\mathcal{R})$.

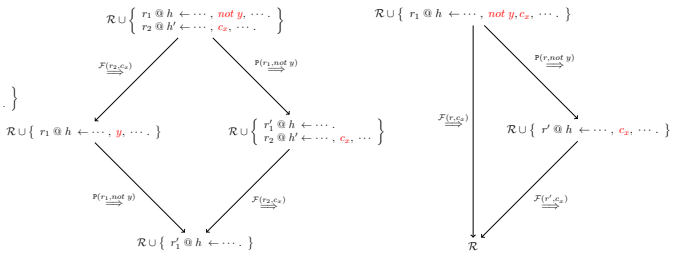

Fig. 8. Confluence of the constraint failure reduction and positive reduction.

The subsequent proof establishes that constraint failure reduction and success reduction are semi-confluent by proving that any divergent reductions from a common origin can eventually be reconciled to a single resulting state. Let $\mathcal{R}$ be a set of rules such that c_x is a satisfied constraint and $y \notin \texttt{heads}(\mathcal{R})$.

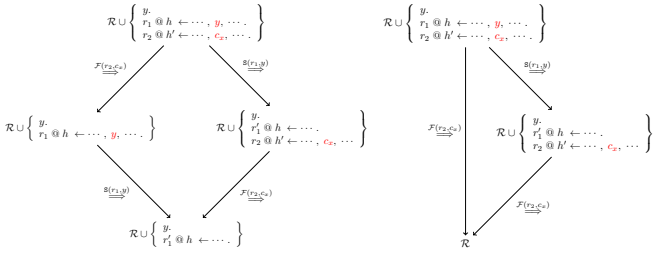

Fig. 9. Confluence of the constraint failure reduction and success reduction.

We recall that the original reductions are proven to be confluent and terminating in [8]. Since every pair of reduction are semi-confluent, the system of reduction is confluent.

Each application of the constraint success reduction removes a constraint atom from the body of a rule. Since each rule

contains a finite number of constraint atoms, the reduction can only be applied a finite number of times to any given program. Consequently, it is impossible to apply the reduction infinitely, ensuring that this reduction will terminate for any given program.

Each application of the constraint failure reduction removes a rule from a program. Since each program contains a finite number of rules, the reduction can only be applied a finite number of times to any given program. Consequently, it is impossible to apply the reduction infinitely, ensuring that this reduction will terminate for any given program.

Proof of Theorem 3. As shown by [8] $\overset{*}{\Longrightarrow}, * \in \{\mathtt{S}, \mathtt{N}, \mathtt{F}, \mathtt{P}, \mathtt{L}\}$ are already confluent and strongly terminating. The filtering reduction does not affect the set of rule of a program, only the domains. This implies that filtering is always confluent with the other reductions. The consistency of filtering also implies that no solution are removed.

Proof of Theorem 4. Direct consequence of Theorem 3 and Theorem 1 of [8].